URZĄD PATENTOWY RZECZYPOSPOLITEJ POLSKIEJ

PCT/PL04/000102

ZAŚWIADCZENIE

Advanced Digital Broadcast Polska Sp. z o.o.
Zielona Góra, Polska

Advanced Digital Broadcast Ltd.
Taipei, Tajwan

złożyły w Urzędzie Patentowym Rzeczypospolitej Polskiej dnia 08 grudnia 2003 r. podanie o udzielenie patentu na wynalazek pt.:**„Sposób emulacji pamięci EEPROM na drodze programowej."**

Dołączone do niniejszego zaświadczenia opis wynalazku, zastrzeżenia patentowe i rysunki są wierną kopią dokumentów złożonych przy podaniu w dniu 08 grudnia 2003 r.

Podanie złożono za numerem P-363945

Warszawa, dnia 31 maja 2004 r.

z upoważnienia **Prezesa**

inż. Barbara Zabczyk

Naczelnik

Sposób emulacji pamięci EEPROM na drodze programowej

Przedmiotem wynalazku jest sposób emulacji pamięci EEPROM na drodze programowej w innej pamięci nieulotnej, przykładowo typu Flash. Sposób ten znajduje zastosowanie w systemach, w których celem obniżenia kosztów urządzeń korzystających z pamięci nieulotnej EEPROM wykorzystuje się istniejącą pamięć, przykładowo typu Flash, do emulacji pamięci EEPROM. Rozwiązanie obejmuje integrację sposobu z układami urządzenia oraz metody zarządzania pamięcią, przykładowo monitorowanie zmian oraz zarządzanie zapisem danych. Dla potrzeb zgłoszenia przestawiono również przykładowy format zapisu danych w emulowanej pamięci.

Z europejskiego zgłoszenia patentowego o numerze EP0991081 jest znany sprzętowy sposób emulacji pamięci w samym układzie scalonym. Jego wadą jest ograniczenie do jednego układu i typu pamięci oraz niski stosunek wielkości emulowanej pamięci EEPROM do wielkości użytej do tego celu pamięci Flash (1:8) oraz mała wielkość emulowanej pamięci, do 8KB. Ujawnione rozwiązanie działa na każdym obecnie stosowanym układzie pamięci Flash, oraz może emulować dowolnie duże obecnie produkowane układy EEPROM, aż do 64KB.

Innym sprzętowym rozwiązaniem jest układ opisany w amerykańskim dokumencie patentowym nr US 5,651,128 przedstawiającym pamięć składającą się z macierzy komórek oraz układów umożliwiających programowanie kasowania za-

wartości emulowanej pamięci. W przykładzie wykonania emulacji korzysta się z pamięci Flash.

W przypadku opisanych rozwiązań emulacji pamięci EEPROM dokonuje się stosując sprzętowe rozwiązania, gdzie układ korzysta z pamięci Flash, aby przechowywać dane zarówno przeznaczone do zapisu w pamięci Flash jak i w pamięci EEPROM.

Różnica pomiędzy przedstawionymi metodami a rozwiązaniem według wynalazku polega na tym, że emulację według wynalazku, w przeciwieństwie do znanych rozwiązań, realizuje się na drodze programowej. Dzięki temu metoda ta może być zastosowana w dowolnym urządzeniu i dowolnym typie pamięci Flash. Istota rozwiązania tkwi w logicznej warstwie programowej, która steruje monitorowaniem i wykorzystaniem emulowanej pamięci. Ponadto wybierając ten rodzaj emulacji obniża się zarówno koszty zakupu układów pamięci jak i zmniejsza się wykorzystanie powierzchni płytek drukowanych eliminując, znacznie droższy od pamięci Flash (w przeliczeniu na 1 KB pamięci), układ pamięci EEPROM, który zwykle montuje się jako osobny moduł. Dodatkowy zysk polega na obniżeniu kosztu oraz na uproszczeniu i miniaturyzacji układu elektronicznego, wykorzystującego dotąd pamięć EEPROM.

W opisanym rozwiązaniu według wynalazku został również przedstawiony przykładowy format zapisu danych w emulowanej pamięci.

Opisany poniżej przykład dekodera telewizji cyfrowej, w którym zastosowano rozwiązanie według wynalazku, należy traktować jako jedno z możliwych zastosowań sposobu emulacji pamięci EEPROM według wynalazku. Każde urządzenie wymagające pamięci nieulotnej pierwszego typu, przykładowo pamięci typu Flash oraz pamięci EEPROM, może zostać zaprojektowane tak, aby wykorzystać spo-

sób emulacji pamięci EEPROM w pamięci pierwszego typu według wynalazku i tym samym obniżyć zarówno koszty układów jak i ilość wymaganego miejsca na układy cyfrowe montowane na płytach drukowanych. Pozwala to na zaprojektowanie urządzeń uniwersalnych, w których kluczowym elementem jest niezależność od konfiguracji bloku pamięci, w którym może wystąpić kombinacja pamięci Flash i EEPROM lub pamięć Flash z emulowaną pamięcią EEPROM. Eliminacja pamięci EEPROM jest zwykle możliwa bez potrzeby zwiększania rozmiaru pamięci FLASH. Zwiększenie pamięci Flash jest bardziej korzystne niż koszt dodatkowego układu EEPROM, z uwagi na cenę pamięci EEPROM wyższą około 32 razy za 1KB pojemności w stosunku do ceny 1KB pamięci FLASH. Dodatkowa pamięć Flash może służyć nie tylko do emulacji pamięci EEPROM.

Jednym z problemów, jakie napotyka się przy emulacji pamięci EEPROM w pamięci typu Flash jest fakt, że pamięć Flash działa w inny sposób. Dane należy zmieniać sektorami, nie można zmieniać danych przykładowo bajtami jak w przypadku pamięci EEPROM, co wymusza zastosowanie sterownika, który udostępniając działanie zgodne z typowymi pamięciami typu EEPROM, będzie operował na sektorach pamięci Flash. Aby zmniejszyć ilość wymaganych zapisów do pamięci Flash, zastosowano kolejne rozwiązanie. Ze względu na to, że podczas typowej pracy pamięci EEPROM dane są aktualizowane często i w małej ilości, przykładowo po jednym bajcie, przy emulacji pamięci EEPROM dane są zbierane i zapisywane po pewnym czasie jako pakiet danych aktualizacyjnych. Czas taki można określić przykładowo w sekundach lub jako ilość wykonanych zmian na danych przechowywanych w pamięci operacyjnej. Dodatkowo zapisywane dane mogą zostać poddane kompresji, jeśli będzie to korzystne. Jednym z wymogów działania emulacji pamięci nieulotnej EEPROM jest zagwarantowanie posiadania

prawidłowej kopii danych nawet, jeśli nie są to najbardziej aktualne dane. Przykładowo taka sytuacja ma miejsce w przypadku zaniku napięcia podczas operacji zapisu danych. Aby zapewnić wymagane bezpieczeństwo danych emulowanej pamięci EEPROM zastosowano obsługę kopii danych. Dlatego system wymaga podwójnej ilość miejsca w pamięci Flash, aby emulować daną ilość pamięci EEPROM. Po zaprogramowaniu danych w jednym z dwóch sektorów pamięci Flash, drugi sektor musi być wykasowany. W przypadku emulacji pamięci EEPROM o wielkości 32KB, wykorzystuje się 2 sektory pamięci Flash, które typowo mają wielkość 64KB każdy. Dodatkowo istnieją trzy typy buforów w pamięci operacyjnej RAM. Pierwszy z nich przechowuje zawsze najbardziej aktualny obraz emulowanej pamięci EEPROM. Drugi przechowuje ostatni pakiet danych aktualizacyjnych, a ostatni opcjonalnie jest używany do przechowywania pakietu danych aktualizacyjnych po kompresji.

Wynalazek jest przedstawiony w przykładzie realizacji na rysunku, na którym fig. 1 przedstawia przykładowe urządzenie w postaci dekodera telewizji cyfrowej wykorzystujące sposób według wynalazku, fig. 2 ujawnia procedurę uruchomienia systemu emulacji pamięci, fig. 3 - procedurę zapisu danych do pamięci, fig. 4 - przykładowy format pakietu danych aktualizacyjnych, fig. 5 przedstawia sektor pamięci Flash zawierający dane emulowanej pamięci EEPROM oraz pakiety danych aktualizacyjnych, fig. 6 obrazuje format nagłówka pakietu danych aktualizacyjnych, fig. 7 - procedurę zapisu danych do aktualizacji, z kolei fig. 8 przedstawia procedurę przygotowania danych do aktualizacji z możliwością unieważnienia poprzednio zapisanego pakietu, a fig. 9 procedurę wyboru aktualnego sektora pamięci nieulotnej.

Przedstawiony na fig. 1 rysunku odbiornik sygnału będący dekoderem telewizji

cyfrowej jest prezentowany dla potrzeb wynalazku w wersji uproszczonej, z ujawnieniem tylko elementów wymaganych dla przedstawienia idei wynalazku. Dekoder telewizji cyfrowej **101** zawiera wiele modułów. Najważniejszym z nich jest procesor **120** zarządzający pracą urządzenia. Dodatkowo według wynalazku, procesor obsługuje wewnętrzny blok **121** sterujący emulacją pamięci EEPROM oraz blok przetwarzania sygnału **122**. Do procesora jest podłączony sygnał z bloku odbioru sygnału **110**. Dodatkowo procesor ma możliwość dwukierunkowej wymiany danych poprzez interfejsy zewnętrzne **140**. Dekoder telewizji cyfrowej zawiera także kilka rodzajów pamięci, które są dwukierunkowo połączone z procesorem. Są to przykładowo pamięć stała, korzystnie typu Flash **150** i operacyjna RAM **160**. W pamięciach tych są przechowywane programy sterujące pracą dekodera telewizji cyfrowej. Bloki **130** oraz **131** umożliwiają odpowiednio nadanie wyjściowego sygnału A/V oraz komunikację z zewnętrznymi urządzeniami kontroli, przykładowo z pilotem.

Na fig. 2 przedstawiono proces inicjowania emulacji pamięci EEPROM według wynalazku. Procedura ta jest wykonywana przy uruchamianiu urządzenia, które stosuje emulację. Procedura rozpoczyna się w punkcie **201**. Następnie, w kroku **202**, wybiera się sektor, z którego będą odczytane dane, jako sektor aktualny. Jeden z dwóch sektorów wybiera się w zależności od kilku kryteriów. Zostaną one szczegółowo przedstawione na fig. 9. Dalej, w kroku **203** procedury, następuje wykasowanie zawartości drugiego sektora. Punkt **203** procedury jest wykonywany, jeśli pomocniczy sektor pamięci zawiera dane, mimo, że nie powinien. Sytuacja taka może mieć miejsce przy pierwszym uruchomieniu emulacji pamięci EEPROM, jeśli w tym sektorze pamięci były wcześniej inne dane lub w przypadku, gdy nastąpiło przerwanie zapisu pierwszego pakietu danych aktualizacyjnych do drugiego sek-

tora, przykładowo wskutek zaniku zasilania lub tuż po jego zakończeniu, ale przed skasowaniem zawartości pierwszego sektora. W takim przypadku, przy kolejnym starcie emulacji istnieją dane w dwóch sektorach i dlatego jeden z nich jest kasowany. Następnie inicjuje się odczyt wybranego sektora **204** pamięci Flash, pobierając do pamięci operacyjnej RAM pierwotny obraz emulowanej pamięci EEPROM. Przykładowo może to być 32KB jednego sektora pamięci Flash, którego wielkość wynosi zwykle 64KB. Jeśli podczas odczytu wystąpi błąd, przykładowo dane nie są poprawne, wtedy jest kasowana zawartość aktualnego sektora. W punkcie **205** procedury pobiera się pierwszy pakiet danych aktualizacyjnych z pamięci Flash i sprawdza czy jest on ważny, czyli czy zawiera dane, które są aktualne **206**. Jeśli pakiet nie został unieważniony (punkt **805**), w punkcie **207** procedury, sprawdza się czy dane pakietu są skompresowane. Jeśli tak, w punkcie **208** jest on dekompresowany. W przeciwnym wypadku procedura przechodzi od razu do punktu **209**, gdzie dane pakietu zapisuje się w pamięci operacyjnej RAM, przechowującej aktualny obraz emulowanej pamięci EEPROM. Ostatnim punktem procedury jest sprawdzenie **210** czy w pamięci Flash są jeszcze dane do odczytania. Jeśli tak, procedura przetwarza kolejne pakiety danych aktualizacyjnych zgodnie z opisanym algorytmem. Jeśli zapisano ostatni pakiet, procedura kończy swoje działanie. Po wykonaniu procedury z fig.2 w pamięci operacyjnej dostępne są wszystkie dane emulowanej pamięci EEPROM.

Na fig.3 przedstawiono procedurę zapisu danych emulowanej pamięci EEPROM w pamięci Flash. Rozpoczyna się ona w punkcie **301**, gdzie przygotowuje się dane do aktualizacji zawartości pamięci Flash. Dane te są także zapisywane w buforze pamięci operacyjnej RAM, która zawsze przechowuje aktualną zawartość emulowanej pamięci EEPROM, co zostanie szczegółowo przedstawione na fig. 8 rysun-

ku. Następnie, w kroku **302** procedury, sprawdza się czy rozmiar pakietu danych aktualizacyjnych jest większy niż ustalona wartość. W prezentowanym przykładzie wykonania są to 64 bajty. Jeśli tak, procedura przechodzi do punktu **303**, gdzie pakiet danych aktualizacyjnych jest poddawany kompresji. W przykładzie implementacji zakłada się, że pakiet danych aktualizacyjnych, którego wielkość jest mniejsza niż 64 bajty, nie jest poddawany kompresji za względu na niskie prawdopodobieństwo osiągnięcia redukcji wielkości pakietu. Następnie, w punkcie **304** procedury, sprawdza się wynik kompresji z punktu **303**. Sprawdzenie to określa czy zysk z kompresji danych jest wystarczająco duży, aby ponieść dodatkowy koszt czasu potrzebnego na dekompresję pakietu w momencie inicjowania systemu emulacji pamięci EEPROM według wynalazku. Jeśli tak, to dalej przetwarza się pakiet skompresowany, a jeśli nie, to przetwarza się pakiet nieskompresowany. Dalej procedura zapisu przechodzi do punktu **305**, w którym następuje sprawdzenie czy w aktualnym sektorze pamięci Flash jest odpowiednia ilość miejsca do zapisu nowego pakietu danych aktualizacyjnych. Jeśli tak, w punkcie **306** zapisuje się pakiet danych aktualizacyjnych, czyli ostatnio modyfikowane dane, w pamięci Flash - ilustruje to szczegółowo fig. 7 w powiązaniu z fig. 6. W przeciwnym przypadku, w punkcie **307** procedury zmienia się aktualny sektor na drugi. Dalej zapisuje się, jako nowy pierwotny obraz emulowanej pamięci EEPROM, pełen obraz przechowywany w buforze pamięci operacyjnej RAM, czyli aktualny obraz emulowanej pamięci EEPROM, w punkcie **308** procedury. Następnie w punkcie **309** kasuje się zawartość drugiego sektora pamięci Flash, gdyż dopiero w tym miejscu jest pewność, że dane emulowanej pamięci EEPROM nie zostaną utracone. W tym miejscu w nowo wybranym sektorze pamięci Flash jest już dodatkowe miejsce na nowe pakiety danych aktualizacyjnych. Procedura zapisu kończy dzia-

łanie w punkcie **310**.

Na fig.4 przedstawiono przykładowy format pakietu danych aktualizacyjnych, według którego zapisuje się informacje w pamięci Flash. Składa się on z czterech pól, z których pierwsze to nagłówek pakietu **401**, drugie to wielkość grupy danych **402** (pole występujące w przypadku pakietów danych aktualizacyjnych, w których występuje wiele grup danych aktualizacyjnych (ang. Multi block patch)), trzecie to przesunięcie danych względem początkowego adresu **403**. To pole występuje tylko w przypadku nieskompresowanych pakietów danych aktualizacyjnych. Ostatnie pole to same dane **404**. Pakiet danych aktualizacyjnych może zawierać wiele grup danych, z których każda zostaje zapisana pod innym adresem pamięci.

W przypadku, gdy pakiet danych aktualizacyjnych jest skompresowany, nie zawiera pola przesunięcia, zaś samą wartość przesunięcia odczytuje się dopiero po dekompresji pakietu. W przypadku, gdy pakiet zawiera wiele grup danych, wartości pól **403** zawierają przesunięcie względem adresu końcowego poprzedniej grupy danych. W ten sposób tylko pierwsze przesunięcie określa adres bezwzględny, a kolejne wartości to adresy względne wobec poprzedniej grupy danych. Umożliwia to zmniejszenie ilości bitów, na których zapisuje się adresy pamięci.

Na fig. 5 przedstawiono podział sektora pamięci Flash na dwie części. Pierwsza z nich **501** to pierwotny obraz emulowanej pamięci EEPROM, a druga **502** to zestaw pakietów danych aktualizacyjnych tej pamięci. Sektor w pamięci Flash ma rozmiar 64KB, wiec mieści zawsze pełen pierwotny obraz emulowanej pamięci EEPROM, nawet nieskompresowany, jak również do kilku tysięcy pakietów danych aktualizacyjnych (ang. Patch). Pełny obraz zapisywany jest na początku każdego sektora jako pierwotny obraz emulowanej pamięci. Można zapisywać za każdym razem pełny obraz i zmieniać sektor za każdym zapisem, ale rozwiązanie

z wykorzystaniem pakietów danych aktualizacyjnych jest bardziej efektywne, gdyż z reguły pakiety te są znacznie mniejsze od 32KB.

Na fig. 6 przedstawiono format nagłówka pakietu danych aktualizacyjnych z fig. 4. Elementy **601** – **604** odpowiadają **401**- **404**. Nagłówek **601** składa się z 7 części **601a** – **601g**. Bit **601a** to bit startu, zmieniany w momencie rozpoczęcia przygotowania do zapisu pakietu danych aktualizacyjnych. **601b** to bit „poprawnego zapisu wielkości oraz formatu" zmieniany po tym, jak wielkość oraz format danych zostaną prawidłowo zapisane. **601c** to bit „poprawnego zapisu" zmieniany po tym, jak cały pakiet danych aktualizacyjnych zostanie poprawnie zapisany w pamięci Flash. Za pomocą tych trzech bitów podczas odczytu danych **206** można stwierdzić ile kolejnych bajtów w pamięci Flash mogło zostać zmienionych i na podstawie tego określić, gdzie może się znajdować kolejny pakiet danych. Jeśli w dowolnym momencie nastąpi zanik zasilania kolejne pakiety danych mogą być zapisywane bez konieczności kasowania całego sektora zaraz za pakietem, którego zapis nie został zakończony. **601d** to bit zmieniany po tym, jak pakiet danych aktualizacyjnych zostaje unieważniony. Jeśli więc kolejny pakiet danych odtwarzałby stan pamięci EEPROM sprzed poprzedniej aktualizacji, można zamiast zapisu kolejnego pakietu unieważnić poprzedni. **601e** to dwa bity, których wartość definiuje format pakietu danych aktualizacyjnych. Za pomocą dwóch bitów można zdefiniować cztery formaty, jednak w przykładowym rozwiązaniu można zdefiniować 3 formaty, a czwartą możliwą wartość zarezerwować na przyszłe rozszerzenie systemu zgodnie z ideą wynalazku. Przykładowe formaty to:

- 0x00 pojedynczy blok nieskompresowany
- 0x01 pojedynczy blok skompresowany
- 0x10 wiele grup nieskompresowanych danych.

Ostatnim polem nagłówka pakietu danych aktualizacyjnych jest **601f**, w którym określa się całkowitą ilość danych w pakiecie danych aktualizacyjnych, nie uwzględniając wielkości nagłówka. W przypadku pól **601f**, **602** oraz **603** dwa pierwsze bity definiują format, w jakim zapisuje się wartości ilości danych oraz przesunięcia adresu. Wartości tych bitów dla pola **602** określać mogą przykładowo:

- 0x00 pole jest opisane z użyciem 4 bitów.
- 0x01 pole jest opisane z użyciem 8 bitów.
- 0x10 pole jest opisane z użyciem 12 bitów.

Wartości tych bitów dla pól **601f** oraz **603** określać mogą przykładowo:

- 0x00 pole jest opisane z użyciem bitów z aktualnego bajtu.
- 0x01 pole jest opisane z użyciem bitów z aktualnego bajtu oraz kolejnego bajtu.
- 0x10 pole jest opisane z użyciem bitów z aktualnego bajtu oraz kolejnych dwóch bajtów.

Na fig. 7 przedstawiono szczegółowo punkt **306** procedury z fig. 3 rysunku. Zapis pakietu danych aktualizacyjnych rozpoczyna się w punkcie **701**, gdzie zmienia się wartość bitu startu **601a**. Następnie w punkcie **702** procedury zapisuje się wielkość oraz typ pakietu danych aktualizacyjnych - pola **601e**, **601f**. Jeśli wystąpi błąd procedura kończy działanie. W większości przypadków błąd, to zanik zasilania. Jeśli zapis wartości pól będzie poprawny, procedura przechodzi do punktu **703**. W kroku tym, zmienia się wartość bitu w polu **601b**. Dalej, w punkcie **704** procedury, zapisuje się poszczególne grupy danych. Jeśli wystąpi błąd procedura kończy działanie. Jeśli zapis będzie poprawny, procedura przechodzi do punktu **705**. W kroku tym, zmienia się wartość bitu w polu **601c**. W tym momencie procedura kończy działanie, a pakiet danych aktualizacyjnych jest poprawnie zapisany

w pamięci Flash. Po dokonaniu zapisu istnieje jeszcze dodatkowa możliwość jego unieważnienia. Jeśli pakiet danych aktualizacyjnych ma zostać unieważniony, w procedurze przedstawionej na fig. 8 zmienia się wartość pola **601d**.

Na fig. 8 przedstawiono procedurę przygotowania danych do aktualizacji z możliwością unieważnienia poprzednio zapisanego pakietu. Diagram jest uszczegółowieniem punktu **301** z fig. 3 rysunku. Procedura rozpoczyna się w punkcie **801**, gdzie pakiet danych aktualizacyjnych jest przygotowywany do zapisu w pamięci. Między innymi jest tworzony nagłówek pakietu. Następnie w punkcie **802** zapisuje się dane w pamięci RAM, która przechowuje najbardziej aktualny obraz pamięci EEPROM, jednocześnie w buforze w pamięci RAM przechowuje się przygotowany pakiet danych aktualizacyjnych. Z kolei w kroku **803** procedury sprawdza się czy ostatnio zapisany w pamięci Flash pakiet danych aktualizacyjnych jest ważny - wartość pola **601d**. Jeśli pakiet jest unieważniony kontynuuje się proces zapisu **807** nowego pakietu danych aktualizacyjnych. W przeciwnym wypadku, gdy pakiet poprzednio zapisany jest ważny, procedura przechodzi do punktu **804**, gdzie sprawdza się czy aktualnie przetwarzany pakiet odwraca zmiany wprowadzone zapisem poprzedniego pakietu danych aktualizacyjnych. Jeśli nie, kontynuuje się proces zapisu **807** nowego pakietu danych aktualizacyjnych. W przeciwnym przypadku, gdy pakiet odwraca zmiany wprowadzone zapisem poprzedniego pakietu, procedura przechodzi do punktu **805**. W tym miejscu zmienia się wartość bitu **601d**, który unieważnia poprzednio zapisany pakiet. Dalej procedura przechodzi do punktu **806**, gdzie anuluje (przerywa) się zapis nowego pakietu danych aktualizacyjnych do pamięci Flash.

Na fig. 9 przedstawiono szczegółowo procedurę wyboru aktualnego sektora, która jest wywoływana w punkcie **202** z fig. 2 rysunku. Aktualny i pomocniczy sektor jest

wybierany na podstawie analizy kilku kryteriów. Pierwsze z nich, to czy poprawnie zakończono poprzedni zapis danych. Drugie, to czy sektor zawiera dane zgodne z wymaganym formatem. Trzecia z możliwości występuje w przypadku, gdy emulacja pamięci EEPROM jest inicjowana po raz pierwszy w danych urządzenia. Procedura rozpoczyna działanie w punkcie **901**, gdzie ustawia się pierwszy sektor jako aktualny. Następnie, w punkcie **902** procedury, sprawdza się czy zapisane dane mają poprawny format oraz czy dane zostały zapisane poprawnie. Można to określić analizując odpowiednie bity nagłówka pakietu danych. Jeśli sprawdzenie z kroku **903** określi, że w sektorze pierwszym znajdują się niepoprawne dane, jako aktualny sektor ustawia się sektor drugi, a sektor pierwszy jako pomocniczy **909**. Jeśli sprawdzenie z kroku **903** określi, że w sektorze pierwszym znajdują się poprawne dane, w punkcie **904** ustawia się drugi sektor jako aktualny. Następnie, w punkcie **905** procedury, sprawdza się czy zapisane dane mają poprawny format oraz czy dane zostały zapisane poprawnie. Jeśli sprawdzenie z kroku **906** określi, że w sektorze drugim znajdują się niepoprawne dane, jako aktualny sektor ustawia się sektor pierwszy, a sektor drugi jako pomocniczy **908**. Jeśli sprawdzenie z kroku **906** określi, że w sektorze pierwszym znajdują się poprawne dane, w punkcie **907** procedury, jako aktualny sektor ustawia się sektor, w którym jest więcej wolnego miejsca. Sektor, w którym jest więcej wolnego miejsca zawiera bardziej aktualne dane. Procedura, wyboru aktualnego oraz pomocniczego sektora kończy działanie w punkcie **910**.

RZECZNIK PATENTOWY
mgr inż. Andrzej Masłowski

Zastrzeżenia patentowe

1. Sposób emulacji pamięci EEPROM w innej pamięci nieulotnej na drodze programowej, **znamienny tym**, że po zainicjowaniu emulacji rezerwuje się dwa sektory pamięci nieulotnej pełniące funkcję sektora aktualnego oraz pomocniczego i tworzy się dwa bufory w pamięci operacyjnej, z których pierwszy przechowuje zawsze najbardziej aktualny obraz emulowanej pamięci EEPROM, a drugi przechowuje ostatni pakiet danych aktualizacyjnych, ponadto aktualny sektor pamięci nieulotnej jest zorganizowany tak, że część sektora zawiera pierwotny obraz emulowanej pamięci, a pozostała część jest sukcesywnie wypełniana pakietami danych aktualizacyjnych opisujących zmiany w zawartości pierwotnego obrazu emulowanej pamięci, z kolei w momencie, gdy nowy pakiet danych aktualizacyjnych nie może zostać dopisany do aktualnego sektora, zamienia się funkcje sektorów pamięci nieulotnej, przez co aktywuje się poprzednio pomocniczy sektor pamięci nielotnej zapisując aktualny obraz emulowanej pamięci z pamięci RAM do nowo aktywowanego sektora jako nowy pierwotny obraz emulowanej pamięci, natomiast po poprawnym zapisie kasuje się zawartość poprzednio aktualnego sektora pamięci nieulotnej.

2. Sposób emulacji pamięci EEPROM według zastrz. 1, **znamienny tym**, że tworzy się trzeci bufor w pamięci operacyjnej, który jest używany do przechowywania pakietu danych aktualizacyjnych po kompresji.

3. Sposób emulacji pamięci EEPROM według zastrz. 2, **znamienny tym**, że przy inicjowaniu emulacji wybiera się sektor aktualny, następnie kasuje się zawartość sektora pomocniczego, a dalej pobiera się z aktualnego sektora pamięci nieulotnej do pamięci operacyjnej RAM pierwotny obraz emulowanej pamięci EEPROM, po czym pobiera się pierwszy pakiet danych aktualizacyjnych z aktualnego sektora pamięci nieulotnej i sprawdza czy jest on ważny, z tym, że jeśli pakiet nie został unieważniony, sprawdza się czy dane pakietu są skompresowane, i w przypadku, w którym nie są one skompresowane zapisuje się je w buforze pamięci operacyjnej RAM, przechowującym najbardziej aktualny obraz emulowanej pamięci EEPROM, natomiast w przypadku przeciwnym przed zapisem do bufora pamięci pakiet danych aktualizacyjnych jest dekompresowany, przy tym w przypadku, gdy pakiet został unieważniony zostaje on pominięty, zaś w ostatnim punkcie procedury inicjowania emulacji sprawdza się czy w pamięci nieulotnej są jeszcze dane do odczytania i jeśli tak przetwarza się kolejne pakiety danych aktualizacyjnych zgodnie z opisanym algorytmem.

4. Sposób emulacji pamięci EEPROM według zastrz. 3, **znamienny tym**, że aktualny i pomocniczy sektor wybiera się ustawiając pierwszy sektor jako aktualny, następnie sprawdza się czy zapisane dane mają poprawny format oraz czy dane zostały zapisane poprawnie, i jeśli w wyniku tego sprawdzenia okaże się, że w sektorze pierwszym znajdują się niepoprawne dane, jako aktualny sektor ustawia się sektor drugi, zaś sektor pierwszy ustawia się jako pomocniczy, natomiast w przypadku przeciwnym, to jest, gdy w sektorze pierwszym znajdują się poprawne dane ustawia się drugi sektor jako aktualny, a następnie sprawdza się czy zapisane dane mają poprawny format oraz czy dane zostały zapisane poprawnie, i jeśli w wyniku tego sprawdzenia okaże się, że w sektorze drugim znajdują się

niepoprawne dane, jako aktualny sektor ustawia się sektor pierwszy, natomiast sektor drugi ustawia się jako pomocniczy, zaś w przypadku przeciwnym jako aktualny sektor ustawia się sektor, w którym jest więcej wolnego miejsca.

5. Sposób emulacji pamięci EEPROM według zastrz. 2, **znamienny tym**, że proces zapisu nowego pakietu danych aktualizacyjnych rozpoczyna się od przygotowania danych do aktualizacji zawartości pamięci nieulotnej, przy czym dane są także zapisywane w buforze pamięci operacyjnej RAM, a następnie sprawdza się czy rozmiar pakietu danych aktualizacyjnych jest większy niż ustalona wartość i jeśli tak, to pakiet danych aktualizacyjnych poddaje się kompresji po czym sprawdza się czy wynik kompresji odpowiada wymaganym założeniom redukcji wielkości pakietu, z tym, że w przypadku zgodności dalej przetwarza się pakiet skompresowany, zaś w przypadku przeciwnym przetwarza się pakiet nieskompresowany, następnie sprawdza się, czy w aktualnym sektorze pamięci nieulotnej jest odpowiednia ilość miejsca do zapisu nowego pakietu danych aktualizacyjnych, i w przypadku wystarczającej ilości miejsca zapisuje się pakiet danych aktualizacyjnych, a w przypadku braku odpowiedniej ilości wolnej przestrzeni zmienia się aktualny sektor na drugi i zapisuje się w drugim sektorze pamięci nieulotnej nowy pierwotny obraz emulowanej pamięci EEPROM, po czym kasuje się zawartość drugiego sektora pamięci Flash.

6. Sposób emulacji pamięci EEPROM według zastrz. 5, **znamienny tym**, że w procesie przygotowania danych aktualizacyjnych zawartości pamięci nieulotnej dokonuje się przygotowywania pakietu danych aktualizacyjnych do zapisu w pamięci, następnie zapisuje się dane w pamięci RAM przechowującej najbardziej aktualny obraz pamięci EEPROM, jednocześnie w buforze w pamięci RAM przechowuje się przygotowany pakiet danych aktualizacyjnych, a następnie sprawdza

się czy ostatnio zapisany w pamięci nieulotnej pakiet danych aktualizacyjnych jest ważny, przy czym jeśli pakiet jest unieważniony kontynuuje się proces zapisu nowego pakietu danych aktualizacyjnych, natomiast w przypadku gdy pakiet poprzednio zapisany jest ważny, sprawdza się czy aktualnie przetwarzany pakiet odwraca zmiany wprowadzone zapisem poprzedniego pakietu danych aktualizacyjnych, z tym, że jeśli przetwarzany pakiet nie odwrócił tych zmian kontynuuje się proces zapisu nowego pakietu danych aktualizacyjnych, a w przeciwnym przypadku, gdy pakiet odwraca zmiany wprowadzone zapisem poprzedniego pakietu, zmienia się wartość bitu, który unieważnia poprzednio zapisany pakiet i anuluje się zapis nowego pakietu danych aktualizacyjnych do pamięci nieulotnej.

7. Sposób emulacji pamięci EEPROM według zastrz. 2, **znamienny tym**, że format pakietu danych aktualizacyjnych składa się z czterech pól, z których pierwsze to nagłówek pakietu, drugim polem występującym w przypadku pakietów danych aktualizacyjnych zawierających wiele grup danych jest pole wielkości grupy danych, trzecim występującym tylko w przypadku nieskompresowanych pakietów danych aktualizacyjnych jest pole przesunięcia danych względem początkowego adresu, natomiast ostatnim polem jest pole danych pakietu, przy czym pakiet danych aktualizacyjnych zawiera wiele grup danych, z których każdą zapisuje się pod innym adresem pamięci, a wartości pola przesunięcia grup danych pakietu zawierają przesunięcie względem adresu końcowego poprzedniej grupy danych, z których tylko pierwsze przesunięcie określa adres bezwzględny, natomiast kolejne wartości to adresy względne wobec poprzedniej grupy danych.

8. Sposób emulacji pamięci EEPROM według zastrz. 7, **znamienny tym**, że skompresowany pakiet danych aktualizacyjnych nie zawiera pola przesunięcia, natomiast samą wartość przesunięcia odczytuje się dopiero po dekompresji

pakietu.

9. Sposób emulacji pamięci EEPROM według zastrz. 7, **znamienny tym**, że format nagłówka pakietu danych aktualizacyjnych składa się z bitu startu zmienianego w momencie rozpoczęcia przygotowania do zapisu pakietu danych aktualizacyjnych, bitu poprawnego zapisu wielkości oraz formatu zmienianego po tym, jak wielkość oraz format danych zostaną prawidłowo zapisane, bitu poprawnego zapisu zmienianego po tym, jak cały pakiet danych aktualizacyjnych zostanie poprawnie zapisany w pamięci nieulotnej, bitu unieważnienia zmienianego po tym, jak pakiet danych aktualizacyjnych zostaje unieważniony i pola określającego format pakietu danych aktualizacyjnych oraz pola określającego całkowitą ilość danych w pakiecie danych aktualizacyjnych, nie uwzględniając wielkości nagłówka.

10. Sposób emulacji pamięci EEPROM według zastrz. 9, **znamienny tym**, że format pakietu danych aktualizacyjnych określa się jako jeden z trzech typów, to jest jako format pojedynczej aktualizacji, albo jako wiele grup danych aktualizacyjnych, albo jako pakiet skompresowany.

11. Sposób emulacji pamięci EEPROM według zastrz. 5, **znamienny tym**, że dla formatu nagłówka pakietu danych aktualizacyjnych składającego się z bitu startu zmienianego w momencie rozpoczęcia przygotowania do zapisu pakietu danych aktualizacyjnych, bitu poprawnego zapisu wielkości oraz formatu zmienianego po tym, jak wielkość oraz format danych zostaną prawidłowo zapisane, bitu poprawnego zapisu zmienianego po tym, jak cały pakiet danych aktualizacyjnych zostanie poprawnie zapisany w pamięci nieulotnej, bitu unieważnienia zmienianego po tym, jak pakiet danych aktualizacyjnych zostaje unieważniony i pola określającego format pakietu danych aktualizacyjnych oraz pola określającego całkowitą ilość danych w pakiecie danych aktualizacyjnych, nie uwzględniając wielkości nagłówka,

zapis pakietu danych aktualizacyjnych rozpoczyna się od zmiany wartości bitu startu, po czym zapisuje się wielkość oraz typ pakietu danych aktualizacyjnych i jeśli wystąpi błąd procedura kończy działanie, natomiast jeśli zapis wartości pól będzie poprawny zmienia się wartość bitu w polu poprawnego zapisu wielkości oraz formatu i dalej zapisuje się poszczególne grupy danych, a jeżeli wystąpi błąd procedura kończy działanie, jeśli natomiast dotychczasowy zapis będzie poprawny zmienia się wartość bitu poprawnego zapisu i w tym momencie kończy się działanie procedury a pakiet danych aktualizacyjnych jest poprawnie zapisany w pamięci nieulotnej.

RZECZNIK PATENTOWY
mgr inż. Andrzej Masłowski

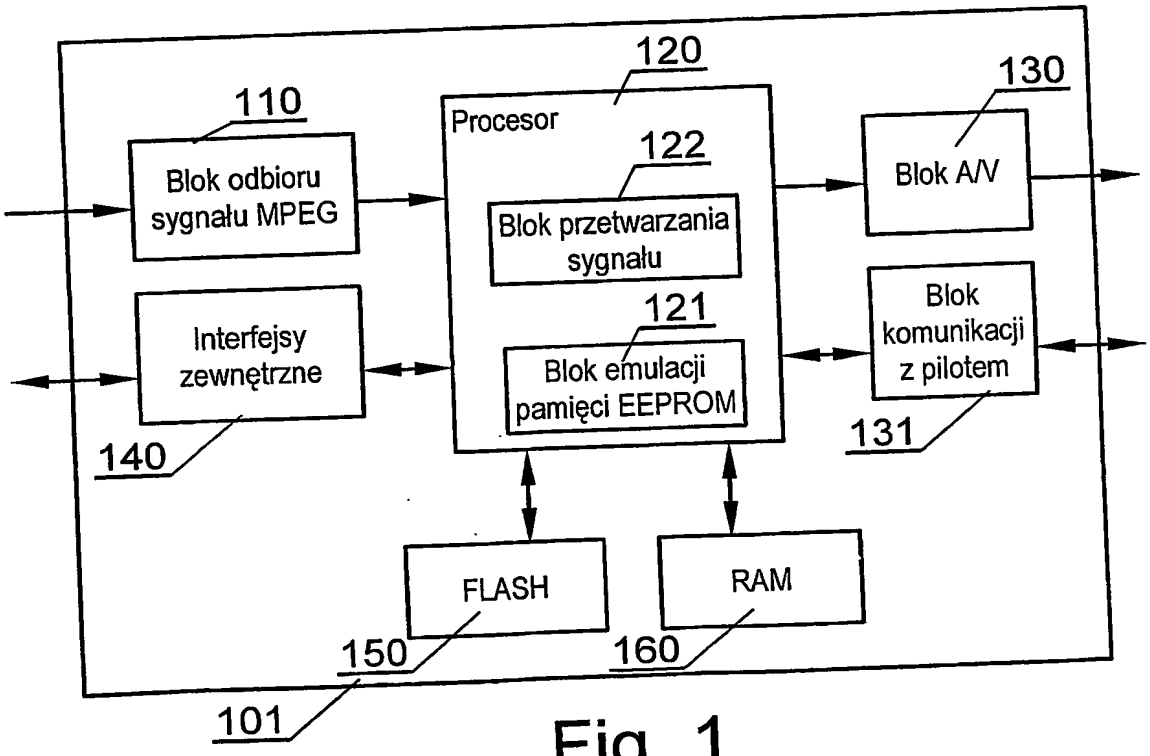


Fig. 1

RZECZNIK PATENTOWY
mgr inż. Andrzej Mieszowski

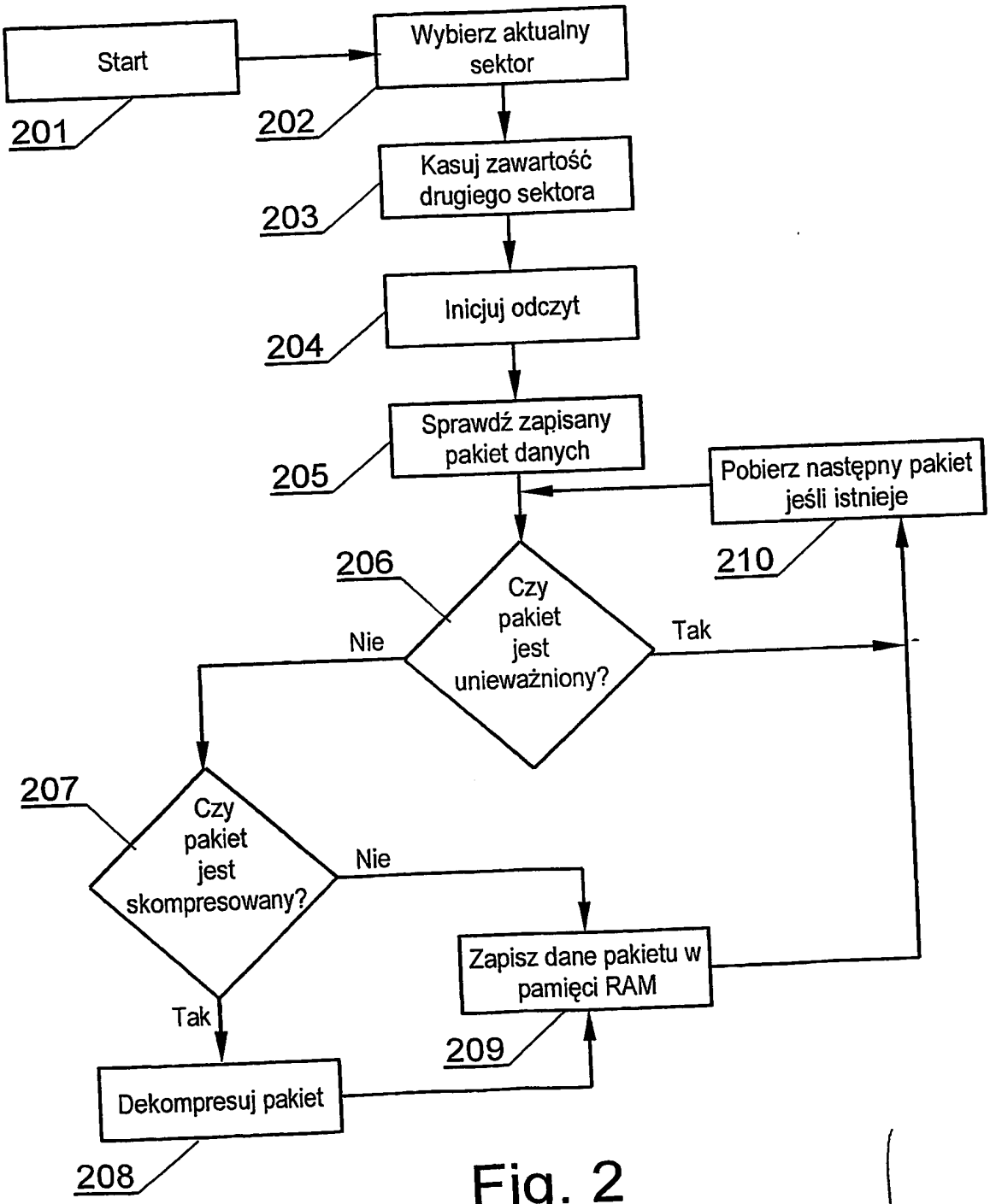


Fig. 2

RZECZNIK PATENTOWY
mgr inż. Andrzej Masłowski

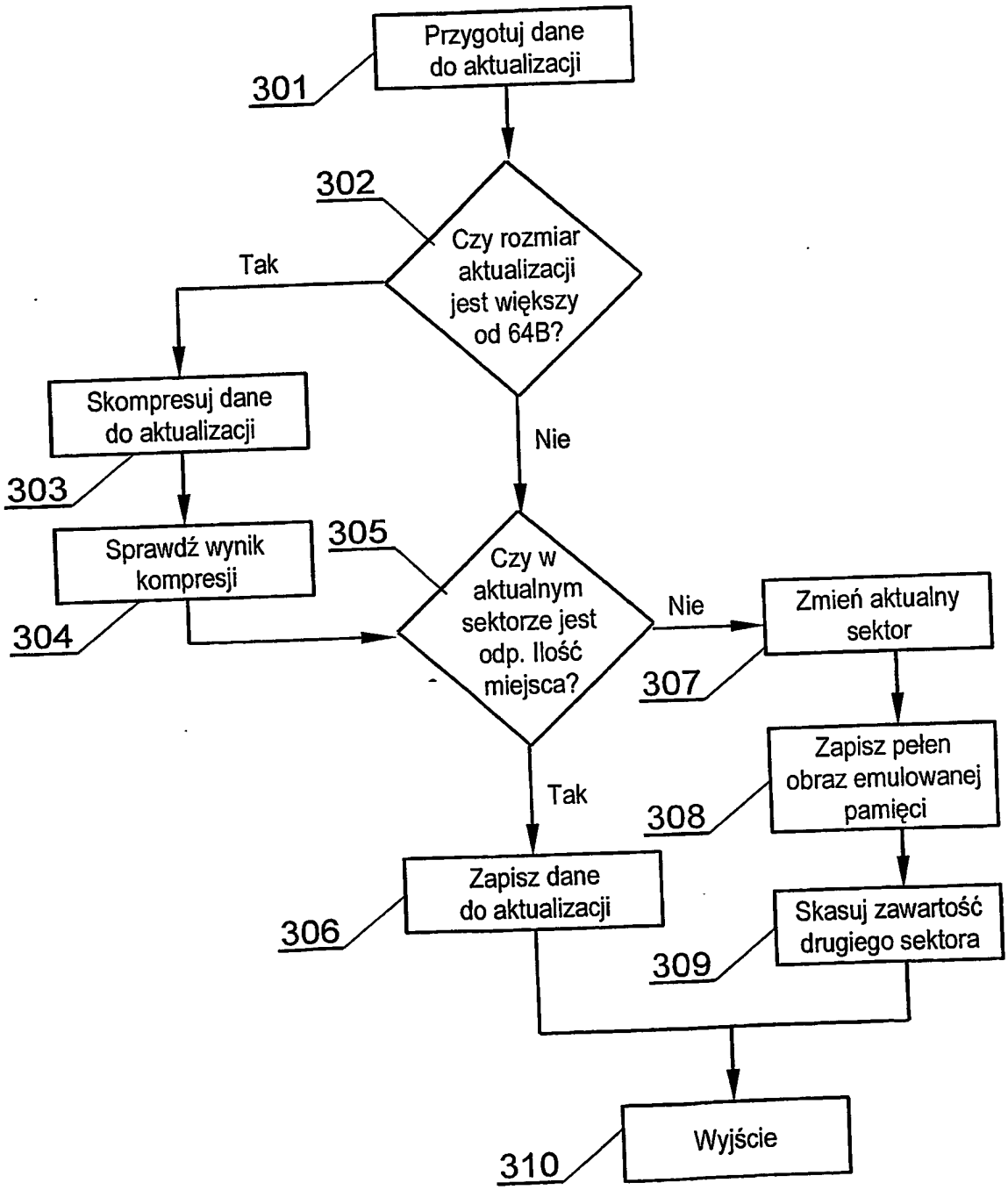

Przygotuj dane do aktualizacji
301
302
Czy rozmiar aktualizacji jest większy od 64B?
Tak
Nie
Skompresuj dane do aktualizacji
303
Sprawdź wynik kompresji
304
305
Czy w aktualnym sektorze jest odp. Ilość miejsca?
Nie
Zmień aktualny sektor
307
Zapisz pełen obraz emulowanej pamięci
308
Tak
Zapisz dane do aktualizacji
306
Skasuj zawartość drugiego sektora
309
Wyjście
310

Fig. 3

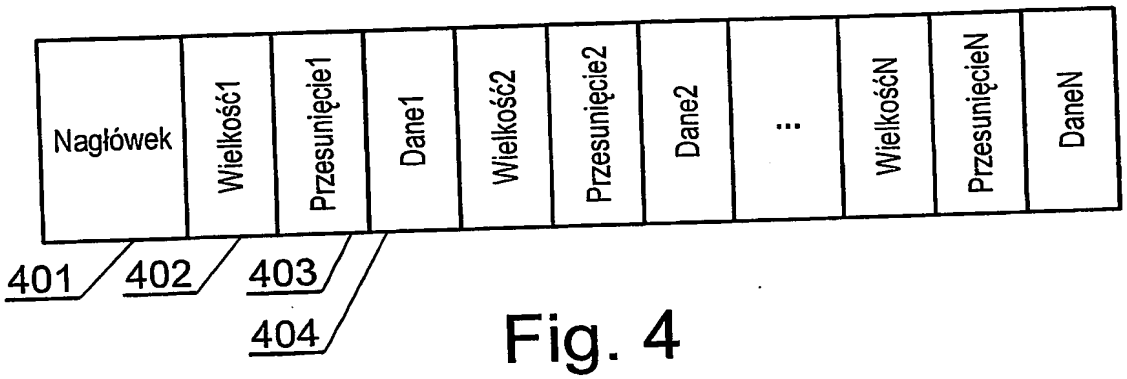


Fig. 4

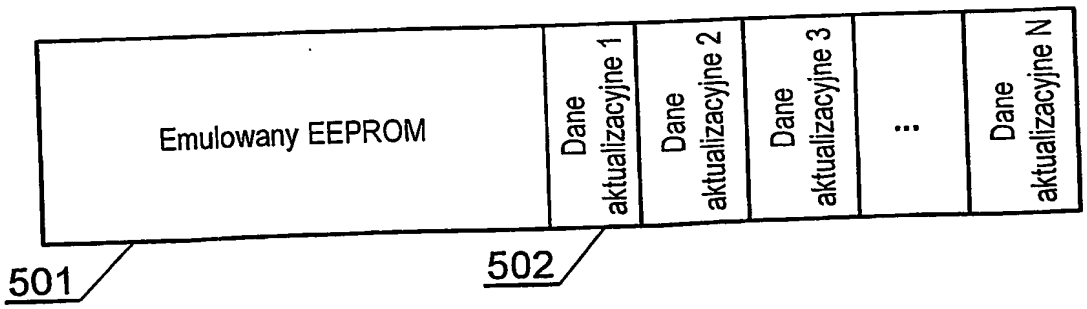


Fig. 5

RZECZNIK PATENTOWY
mgr inż. Andrzej Masłowski

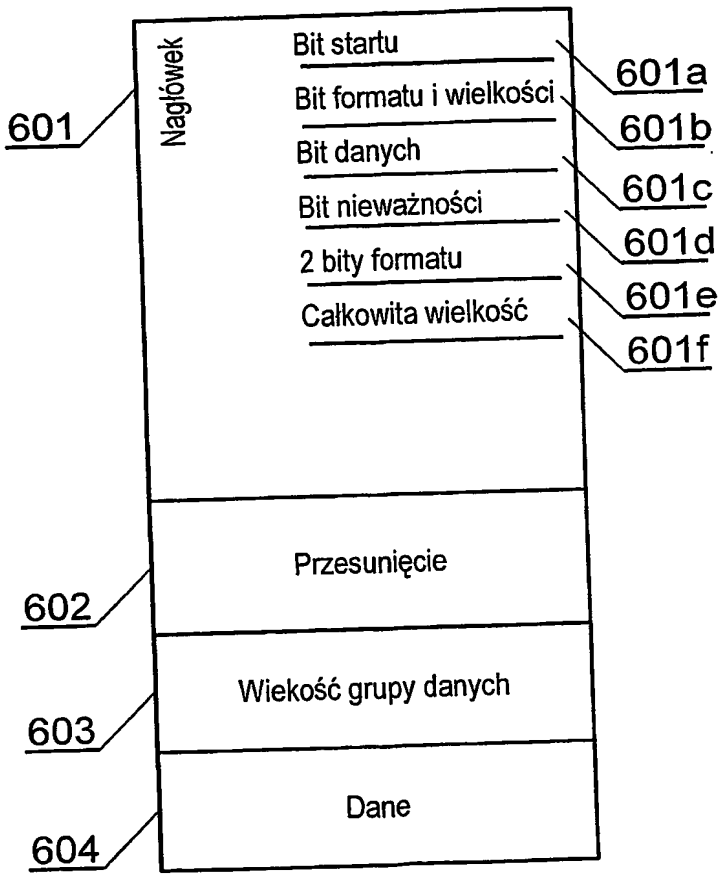
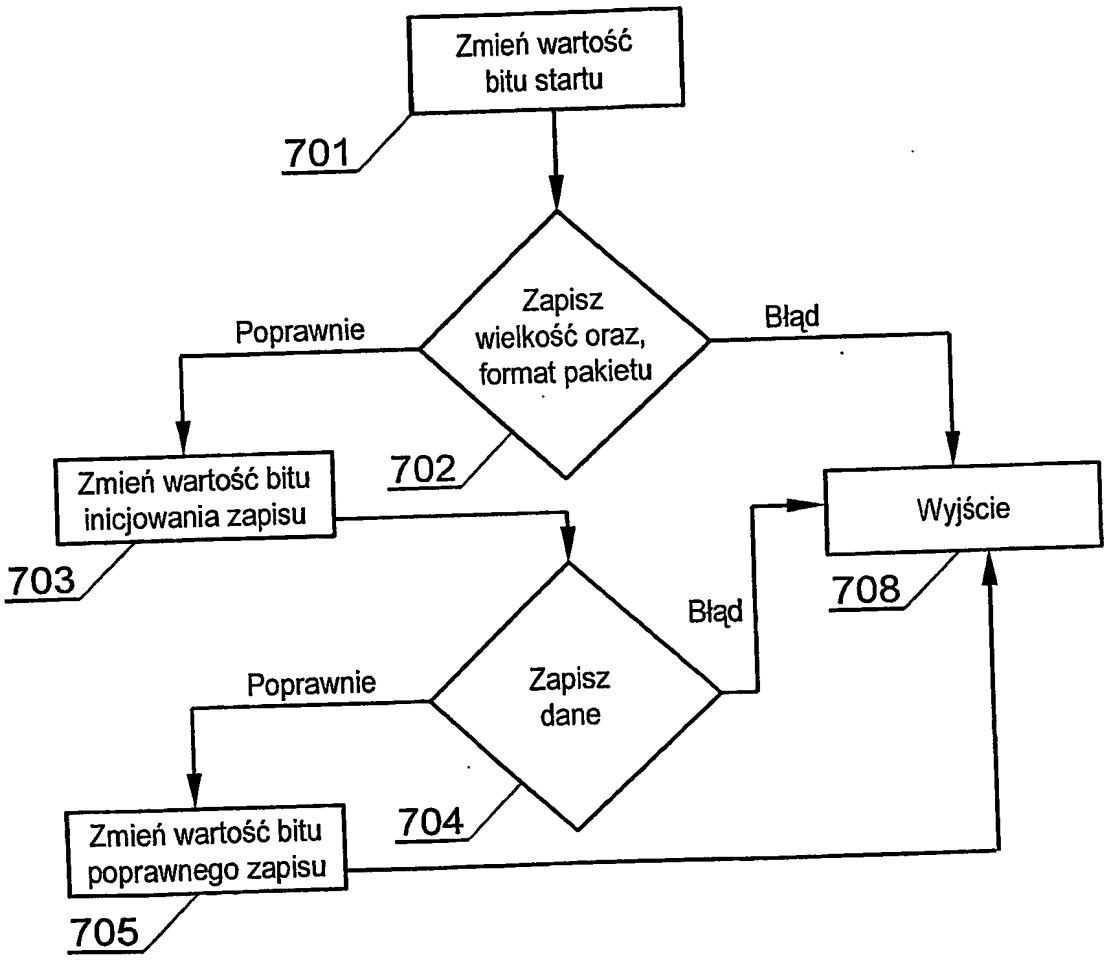


Fig. 6

RZECZNIK PATENTOWY
mgr inż. Andrzej Mazowiecki



RZECZNIK PATENTOWY
mgr inż. Andrzej Maćkowski

Fig. 7

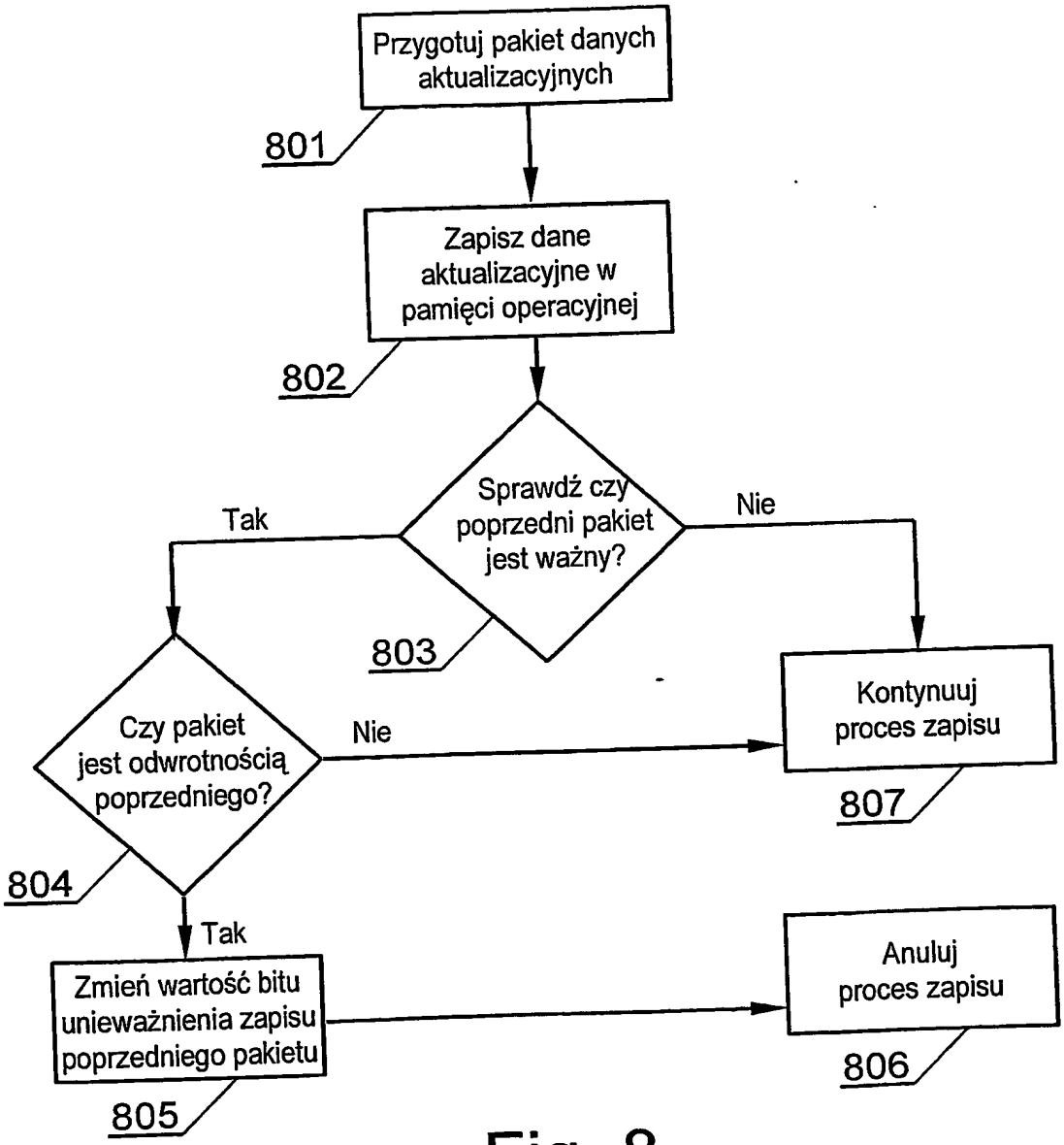


Fig. 8

RZECZNIK PATENTOWY
mgr inż. Andrzej Mazowski

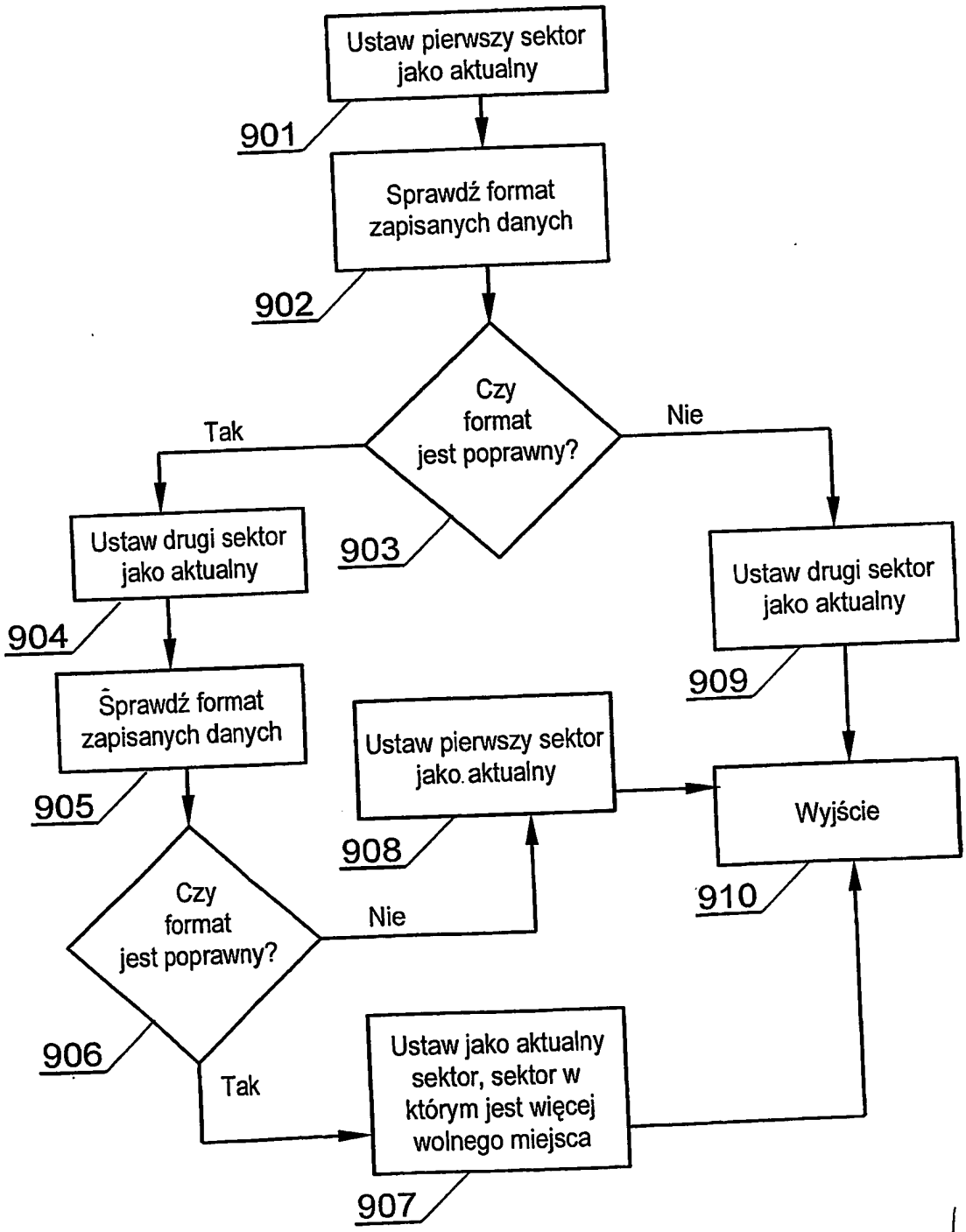


Fig. 9

RZECZNIK PATENTOWY
mgr inż. Andrzej Kacsicwski

Document made available under the Patent Cooperation Treaty (PCT)

International application number: PCT/PL04/000102

International filing date: 06 December 2004 (06.12.2004)

Document type: Certified copy of priority document

Document details: Country/Office: PL
Number: P-363945
Filing date: 08 December 2003 (08.12.2003)

Date of receipt at the International Bureau: 29 March 2005 (29.03.2005)

Remark: Priority document submitted or transmitted to the International Bureau in compliance with Rule 17.1(a) or (b)

World Intellectual Property Organization (WIPO) - Geneva, Switzerland
Organisation Mondiale de la Propriété Intellectuelle (OMPI) - Genève, Suisse

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☒ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☒ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.
As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.